SERVER FOR TRACKERS MANUAL

Version 8.6.0

January 9th, 2013

Copyright © 2013 by NAL Research Corporation
9300 West Courthouse Road
Manassas, Virginia 20110 USA
Phone: 703-392-1136 x203
E-mail: contact@nalresearch.com

LEGAL DISCLAIMER AND CONDITIONS OF USE

This document contains information for the NAL Researcher modems and trackers and accompanying accessories ("Product") is provided "as is." Reasonable effort has been made to make the information in this document reliable and consistent with specifications, test measurements and other information. However, NAL Research Corporation and its affiliated companies, directors, officers, employees, agents, trustees or consultants ("NAL Research") assume no responsibility for any typographical, technical, content or other inaccuracies in this document. NAL Research reserves the right in its sole discretion and without notice to you to change Product specifications and materials and/or revise this document or withdraw it at any time. User assumes the full risk of using the Product specifications and any other information provided.

NAL Research makes no representations, guarantees, conditions or warranties, either express or implied, including without limitation, any implied representations, guarantees, conditions or warranties of merchantability and fitness for a particular purpose, non-infringement, satisfactory quality, non-interference, accuracy of informational content, or arising from a course of dealing, law, usage, or trade practice, use, or related to the performance or nonperformance of any products, accessories, facilities or services or information except as expressly stated in this guide and/or the Product and/or satellite service documentation. Any other standards of performance, guarantees, conditions and warranties are hereby expressly excluded and disclaimed to the fullest extent permitted by the law. This disclaimer and exclusion shall apply even if the express limited warranty contained in this guide or such documentation fails of its essential purpose.

In no event shall NAL Research be liable, whether in contract or tort or any other legal theory, including without limitation strict liability, gross negligence or negligence, for any damages in excess of the purchase price of the Product, including any direct, indirect, incidental, special or consequential damages of any kind, or loss of revenue or profits, loss of business, loss of privacy, loss of use, loss of time or inconvenience, loss of information or data, software or applications or other financial loss caused by the Product (including hardware, software and/or firmware) and/or the Iridium satellite services, or arising out of or in connection with the ability or inability to use the Product (including hardware, software and/or firmware) and/or the Iridium satellite services to the fullest extent these damages may be disclaimed by law and whether advised of the possibilities of such damages. NAL Research is not liable for any claim made by a third party or made by you for a third party.

TABLE OF CONTENTS

1.0 INTRODUCTION 4

2.0 GETTING STARTED 5
2.1 System Requirements 5
2.2 Installation 5
2.3 Service / Controller Interaction 5
2.4 Starting and Stopping the Service 5
2.5 Running the Controller 6

3.0 CONFIGURING THE CONTROLLER 8
3.1 Service Connection 8
3.2 Number List 8
3.3 Appearance 9

4.0 CONFIGURING THE SERVICE 10
4.1 Service 11
4.2 Communication Links 11
4.3 Processing 18
4.4 Output 19

5.0 CONTROLLING THE SERVICE 26
5.1 Contacting Remote Modems 26

6.0 MONITORING THE SERVICE 28
6.1 Messages 28
6.2 Log File 28

7.0 SENDING REMOTE UPDATES 30

8.0 TRACKING REMOTE UPDATES 32

APPENDIX A – XML FORMATS 33
GPS Reports 34
Update Responses 37
Status Report 0 44

1.0 INTRODUCTION

Iridium satellite trackers developed by NAL Research are stand-alone units each relying on built-in controllers for operation instead of an external Data Terminal Equipment (DTE). As satellite trackers, their primary function is to send tracking reports and emergency alerts to a network operating center (NOC) at pre-programmed interval. While in the field, they can also accept real-time instructions/commands from a NOC to change their operating parameters.

A NOC could be a sophisticated command/control facility equipped with servers connected to high-speed networks. A NOC could also be as simple as a smart phone, a laptop or a desktop connected to the Internet. NAL Research provides Server for Trackers with the intent of allowing users to quickly install on their NOCs the software needed for receiving tracking reports and sending operating parameters to remote trackers.

Server for Trackers has many features that also make it a good solution for communicating with other non-tracker satellite modems developed by NAL Research such as the A3LA-RM. In particular, the software handles encryption and decryption, implements packet communication with the A3LA-RM through a RUDICS connection, opens a connection pipe to a socket for sending and receiving data, and provides a rich set of logging capabilities.

In order to make the program more reliable, Server for Trackers is composed of two components—a Windows® service and a GUI controller program. The service runs in the background and handles communication with the remote modems according to its settings. The controller acts as the user interface to the service and allows the user to configure, monitor, and control the service. At any time, the controller can be closed and the service continues to run. Also, if the computer is rebooted (for example due to a power outage), the service will startup automatically.

2.0 GETTING STARTED

2.1 System Requirements

Before installing the Server for Trackers software, make sure the NOC server computer meets the following requirements:

- Compatible with Microsoft .NET 3.5
- Minimum of 8MB available on hard drive
- 800 x 600 screen resolution (looks best with a screen resolution of at least 1024 x 768)

2.2 Installation

The CD-ROM provided by NAL Research has two Server for Trackers Setup programs—a setup program for the Server for Trackers service and a setup program for the Server for Trackers controller. Both the service and controller are needed. To install, double click on the setup programs and follow the prompts.

The service and controller may be installed on separate computers if needed. With this setup, in order for the controller to communicate with the service, the C:\ drive of the computer with the service needs to be mapped to a drive letter on the computer with the controller. Also, the controller needs to be setup to point to the service (see the section on configuring the controller).

2.3 Service / Controller Interaction

The service and controller interact via a socket connection, which is referred to as the control server, and also through files, which both the service and the controller have access to. These files include the service settings file, the service log file, the remote update pairs file, and the Crypto Officer files.

The control server connection is used by the controller mainly to send commands to the service whereas the service uses the control server connection mainly to keep the GUI of the controller up to date.

For the most part, the communication between the service and controller via common files is accomplished by the controller writing the files and the service reading them. This is the case for the service settings file and the Crypto Officer files. However, the remote update pairs file and the service log file are written by the service and read by the controller.

With the way that the service and controller interact, it is not necessary for the service to know the location of the controller since it plays the role of the server in the control server connection and because all of the common files are stored in the same folder as the other service related data. However, the controller needs to know the directory that contains the common files so that it can access the common files. The controller also needs to know the host IP address of the computer that the service is installed on as well as the control server port so that it can connect to the control server.

2.4 Starting and Stopping the Service

After installation completes, the service will be installed with a startup type of "Automatic" but will not yet be started. A startup type of "Automatic" means that the service will start automatically whenever the computer reboots.

Windows provides a services administration tool that can be used to change the startup type as well as to manually start and stop the service. This tool can be accessed from "Control Panel | Administrative Tools | Services". In the "Services" window, look for the service named "Server for Trackers Service".

Whenever the service is started, it loads its settings from an XML file named "ServiceSettings.xml" in the "NAL\Server for Trackers Service" subfolder of the common application data folder. All of the settings in this file can be updated via the "Configure Service" window of the controller. Based on the settings loaded from this file, the service may also load some additional settings, such as encryption keys, which are also stored in the same subfolder.

2.5 Running the Controller

After the service is installed and started, it will need to be configured before it can receive any reports. For configuring, monitoring, and controlling the service the controller is needed.

The controller can be accessed from the Windows® start menu in the "All Programs | NAL Research" folder. When the controller is opened as shown in Figure 1, its default settings are loaded. The default settings are stored in an XML file named "DefaultControllerSettings.xml" in the "NAL\Server for Trackers Controller" subfolder of the common application data folder. All of the settings in this file can be updated via the "Configure Controller" window.

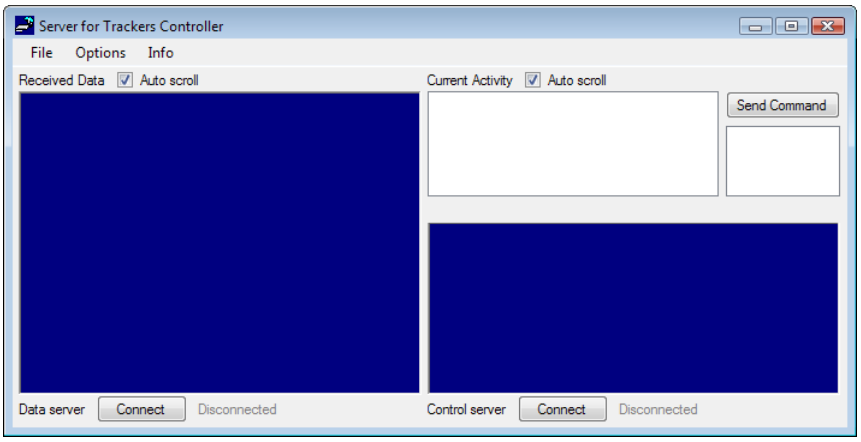


Figure 1. Startup Screen of Server for Trackers Controller.

After starting the controller, it still needs to be connected to the service to enable all of the features of the controller. While not connected to the service, however, the controller can still configure the service, view the log file of the service, and view remote update tracking information since these operations depend only on accessing the common files. To connect the controller to the service, click the control server "Connect" button at the bottom of the window. If the firewall prompts about blocking the controller, choose to "Unblock". After unblocking, the "Connect" button may need to be clicked again. If the service and controller were installed on different computers or the service was not able to listen for connections due to the default control server port not being available, then the service and controller will need to be configured before the controller can connect to the service.

Once the controller and service are connected, the controller will be able to send commands to the service and the service will be able to send information to the controller to keep the GUI up to date. The service will also be able to send messages to the controller to be displayed to the user, such as feedback when the service is configured.

When the controller is no longer needed it may be closed without affecting the service. The service will continue to run as normal. Therefore, the controller can be opened to configure the service and then closed after the configuration is complete. Also, the controller can be opened periodically to check on the status of the service.

Using the Main Screen

The main screen consists of two main sections, a "Received Data" section and a "Current Activity" section. The "Received Data" section is for displaying data received from the data server of the service. The data server is explained in the "Configuring the Service" section. When the controller is connected to the service, the "Current Activity" section lists all of the communication links that are setup. When a communication link is selected, its status and activity along with a list of commands that can be sent to it are displayed. Sending commands to the service is fully covered in the "Controlling the Service" section.

3.0 CONFIGURING THE CONTROLLER

The "Configure Controller" window, which can be opened by choosing the "Options | Configure Controller..." menu item, is the central place where all controller settings are configured. This window has three screens, which are explained in the following sections. Any changes made in this window will take effect immediately in the controller program. However the changes will not be saved until the "Save Settings" button is clicked. If the "Save as default" checkbox is checked, clicking on the "Save Settings" button will write the settings to the controller's default settings file. If "Save as default" is not checked, clicking on the "Save Settings" button will prompt for a file to save to. The "Load Settings" button can be used to load a settings file other than the default.

3.1 Service Connection

Because the controller is the GUI for the service, it needs to know how to connect to the service. The "Service Connection" screen contains all of the settings that the controller needs in order to communicate with the service (Figure 2).

The "Data directory" field in the "Service Location" group box is the directory where the data files for the service are located, while the "Host" field in the same group box is the host name or IP address of the computer that has the service installed. If the service was installed on a different computer, the default settings will need to be updated. The port for the control server and data server should match the cooresponding settings in the service. The "Connect on startup" options are for the convience of not having to click the "Connect" buttons on the main window every time the controller is started.

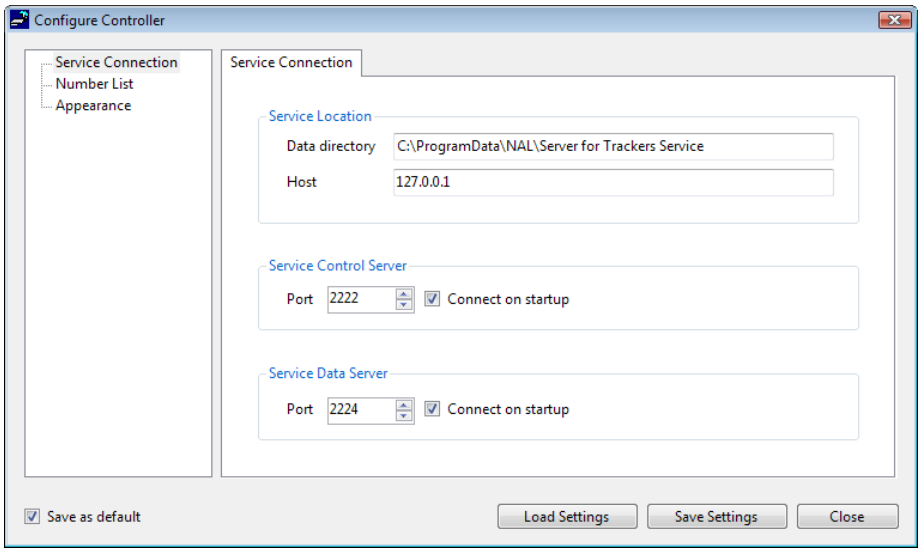


Figure 2. Configure Controller – Service Connection Screen.

3.2 Number List

As the GUI for the service, the controller's settings file is a logical place to store usability settings, such as a number list. The "Number List" screen allows the user to build a list of numbers associated with descriptions for use in other parts of the controller, such as the "Contact Remote Modems" screen. This saves the user from having to retype the same number every time the controller is restarted.

To add a new entry just start typing into the bottom row, to edit any value in the list double click the value, and to delete an entry select the row by clicking in the area to the left of the entry and press the delete key.

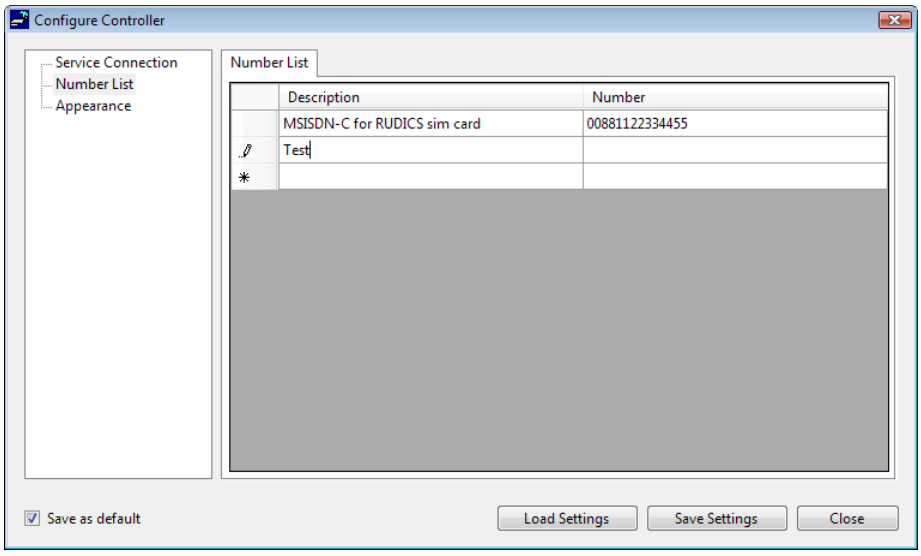


Figure 3. Configure Controller – Number List Screen.

3.3 Appearance

On the "Appearance" screen shown in Figure 4, the font for the main window's "Received Data" and "Current Activity" areas can be changed. Only the font type and size will be saved to the settings file.

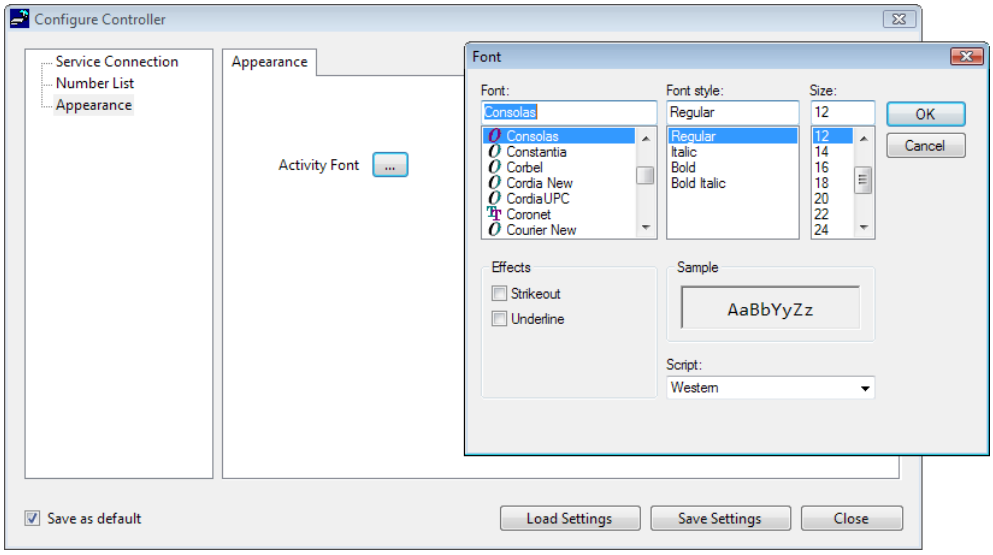


Figure 4. Configure Controller – Appearance Screen.

4.0 CONFIGURING THE SERVICE

All of the settings for the service can be configured with the "Configure Service" window as shown in Figure 5. To open this window, chose "Configure Service..." from the "Options" menu. Whenever the "Configure Service" window is opened, the controller reloads the settings file of the service it is setup for into the window. This is to make sure the window is up to date before any changes are made. To set the controller up for a different service, see the section on configuring the controller. The "Save to" option will show the name of the service's settings file. If this option is selected, then clicking the "Save Settings" button will overwrite the settings file of the service that the controller is setup for. This option can be unselected to allow the user choose a different settings file to overwrite. The "Load Settings" button is provided so that a preconfigured settings file can be loaded into the window.

The way that the controller configures the service is to overwrite the settings file of the service. The service is always monitoring the file for changes. As soon as a change is made to the settings file, the service reloads the file and updates its running configuration. Therefore, with the exception of encryption settings, which are stored in different files, no changes are made to the service until the "Save Settings" button is clicked. Closing the "Configure Service" window before clicking the "Save Settings" button will cause all of the changes made in the window to be lost.

When the service loads the settings file and updates its settings, it logs the changes that are made to the service log file. Also, if the controller is connected to the service, the service will send messages to the controller describing what was changed. The controller will popup the "Monitor Service Status" window to show these messages.

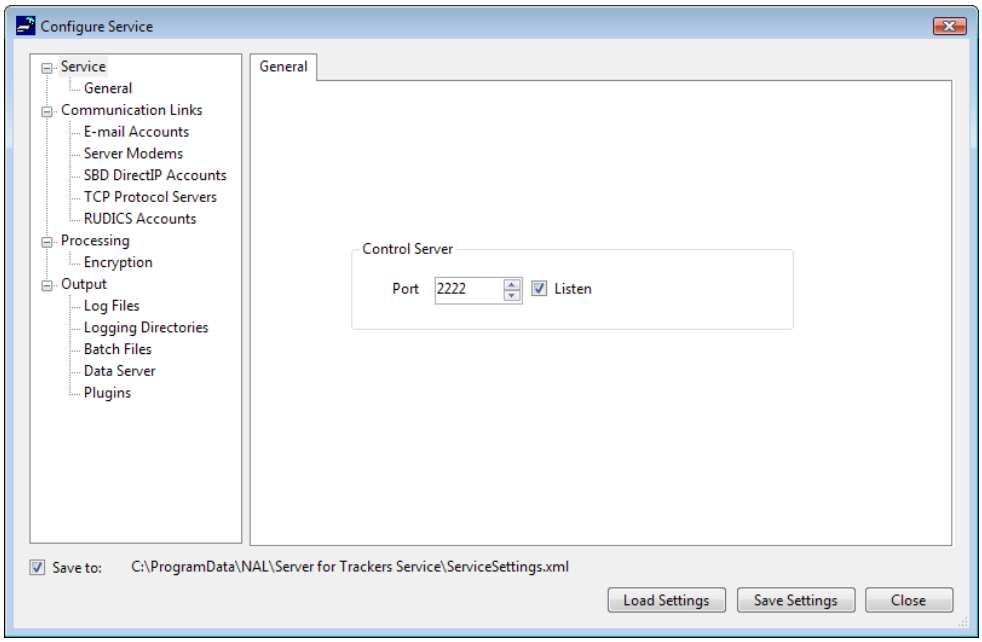


Figure 5. Configure Service – Service General Screen.

4.1 Service

The first group of settings in the "Configure Service" window is the "Service" group. This group only has one screen named "General". In this screen, are the very important control server settings. The control server listens for connections from the controller and only accepts one connection at a time. Before the service is able to receive a connection from the controller, the port should be set to an available port and the listen option should be checked. To disconnect the controller and to stop accepting connections, the listen option should be unchecked.

4.2 Communication Links

Depending on the model of the remote modem, a NOC has five choices to communicate with it—via modems (landline and/or Iridium modems) connected to its computer's serial ports, via e-mail accounts, via SBD DirectIP, via RUDICS, or via TCP connections. The "Communication Links" group in the "Configure Controller" window contains screens for setting up these five types of communication links. The screens have a similar layout and the process of adding, editing, and deleting communication links on each screen is the same. There is a list of current communication links on the left. When a communication link in the list is selected, its settings are shown on the right and can then be modified. There is an "Add" button below the list, which can be clicked to add a new communication link with the default settings. The "Remove" button removes the currently selected link in the list and the "Remove All" button removes all of the links in the list.

The following sections will explain each type of communication link and describe their settings.

E-mail Accounts

Server for Trackers uses the e-mail account communication link to retrieve SBD and SMS messages from a POP3 server and to send remote updates via an SMTP server. Most of the settings on this screen are just typical e-mail accout settings, such as the server, port, user name, and password of the POP3 and SMTP servers.

The "Auto retrieve every ... minutes" option can be set to periodically retrieve emails from the POP3 server. If this option is set at 0 minutes, then the email account will retrieve continuously, meaning that as soon as one retrieve operation is completed, another one will immediately be started. Note that some POP3 servers will disable the account for a period of time due to frequent logins.

If the "Delete from server" option is selected, emails will be deleted from the POP3 server after being retrieved. The "All" and "SBD / SMS only" options determine whether all emails or only emails that contain SBD and SMS messages will be deleted from the server. For emails that are not deleted, Server for Trackers will keep track of the UIDs so as not to retrieve them again. The UIDs are stored on separate lines in a file with the .uidls extension in the "NAL/Server for Trackers Service" subfolder of the common application data directory. Deleting a line in this file will enable the email account to retrieve the cooresponding email again. Note that if too many messages build up on the POP3 server, the retrieval process may be slowed down.

NOTE: Server for Trackers only recognizes SMS email messages from Iridium and T-Mobile. However, future support for T-Mobile is not guaranteed.

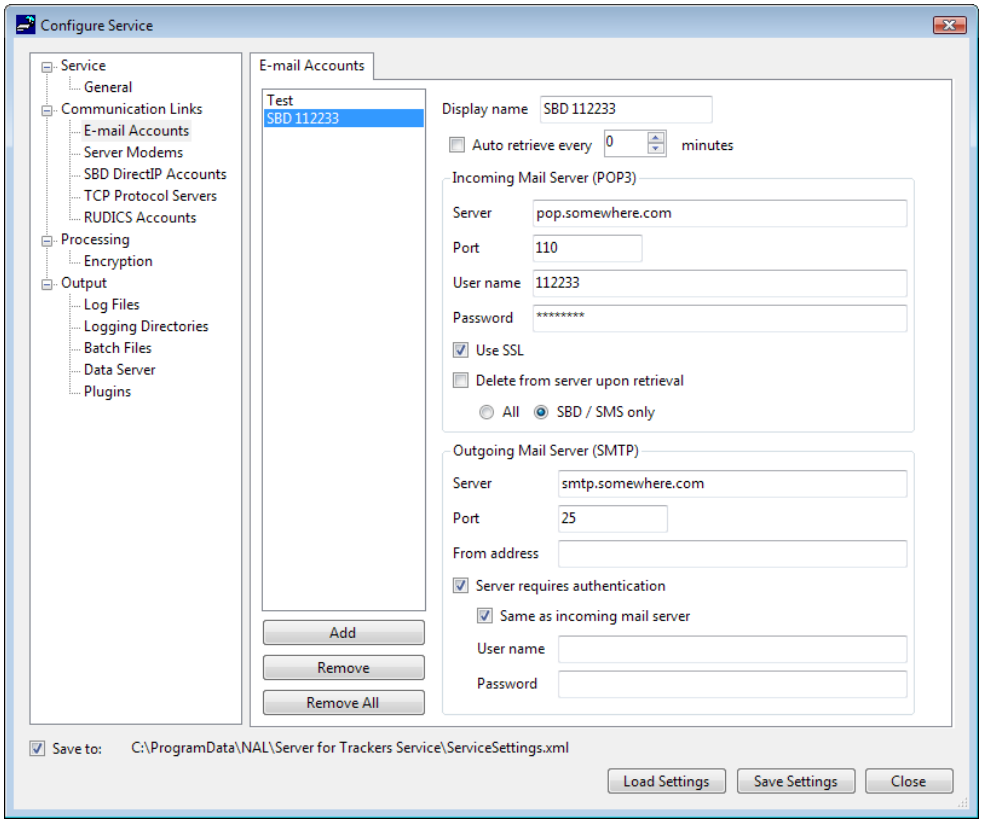


Figure 6. Configure Service – E-mail Accounts Screen.

Server Modems

Server for Trackers uses the server modems to receive and initiate calls as well as to receive SMS messages. All Iridium A3LA modem series and most but not all Hayes-compatible landline modems may be used as a server modem.

The server modem communication link has settings for setting the port that the server modem is connected to, and the baud rate, stop bits, parity, and data bit properties for the port. If the server modem is an Iridium satellite modem, then the baud rate should be set to 19200, stop bits should be set to 1, the parity should be set to "None", and the data bits should be set to 8.

The "Receive SMS messages" option instructs the server modem communication link to set up the server modem to receive SMS messages immediately without storing them in the SMS memory by executing the AT+CMNI=2,2 command. This option is required for server modems to communicate with A3LA-TSS trackers by SMS.

The "Manage connection for satellite mdoems" option instructs the link to check for the satellite signal strength occasionally with the AT+CSQ command. If the satellite connection has been lost, the link will renegotiate the connection with the AT+COPS=1 command. This option should only be used with satellite modems.

If the "Automatically answer calls with ATS0=1" option is selected, the link will use the ATS0=1 command to cause the server modem to automatically answer calls after one ring. This option should be selected if the server modem is to be used to receive data calls.

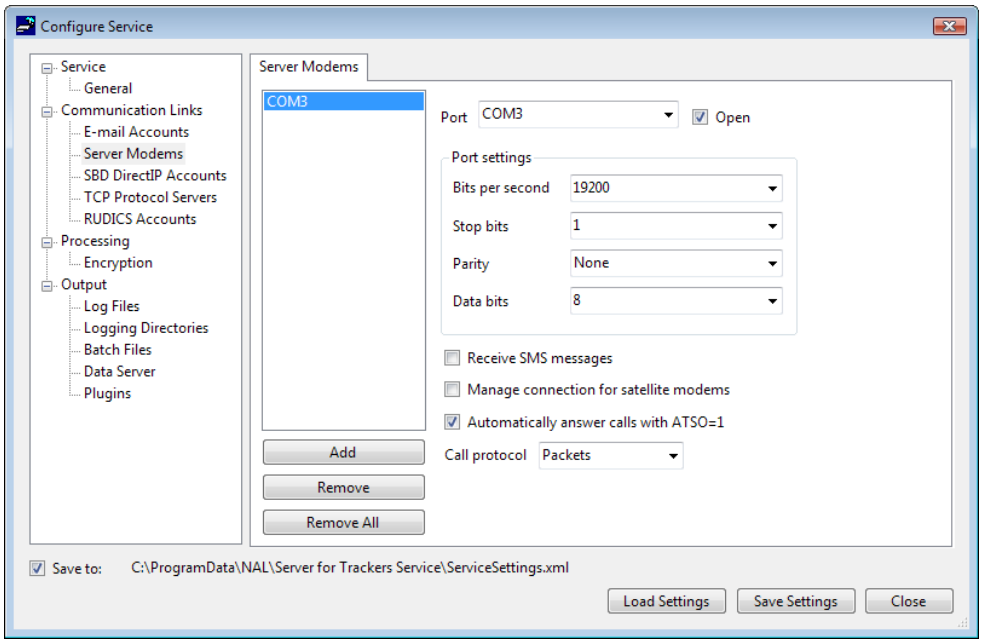


Figure 7. Configure Service – Server Modems Screen.

The "Call protocol" setting determines how data is sent and received from the remote modem. There are two options for the call protocol. The "None" option, means that no protocol will be used and that data will be sent and received as is. The "Packets" protocol means that data will be sent and received in packets. By using packets, data transmission is more reliable allowing even large files to be transferred.

In order for Server for Trackers to receive from or send any data with the server modem, the port needs to be opened by checking the "Open" option and the modem needs to be turned on.

NOTE: Server for Trackers program is supposed to initialize the modem when it starts with ATS0=1V1E1. It sometimes does not send over the initialization string to the connected modem and thus the modem will not automatically answer incoming calls. If a modem is going to be used a lot to receive calls we recommend making the initialization the default setting with the following AT commands:

- ATS0=1V1E1
- AT&W0
- AT&Y0

Understanding and Troubleshooting SMS Commands and Errors

AT+CMNI=2,2 Causes the modem to receive SMS messages immediately without storing them in the SMS memory. If this command returns an error, it may be because the SMS memory is full. Any serial communications program, such as SatTerm (from NAL Research) or HyperTerminal can be used to view/delete messages in the SMS memory. Use the AT+CMGL=4 command to list all of the messages in the SMS

	memory. This command will show the message number followed by the message. Then use the AT+CMGD=# command, where # is the number of the message to delete.
+CMT	Displayed when an SMS message is received.
AT+CMGS	Sends an SMS message.
+CMS ERROR:331	No network service. This usually means that the modem is not getting strong enough signal strength.
+CMS ERROR:300	Phone failure. This does not necessarily mean that the command did not work. However, the program will assume that the command did not work. If this error continues to occur, restarting the modem will most likely get rid of it. If restarting the modem does not fix the problem, try deleting some messages from the SMS memory as described in the description of the AT+CMNI=2,2 command.
+CMS ERROR:302	Operation not allowed. This does not necessarily mean that the command did not work. However, the program will assume that the command did not work. If this error continues to occur, restarting the modem will most likely get rid of it. If restarting the modem does not fix the problem, try deleting some messages from the SMS memory as described in the description of the AT+CMNI=2,2 command.

SBD DirectIP Accounts

A NOC server can communicate with remote satellite trackers using the SBD DirectIP protocol. SBD DirectIP allows SBD messages to be sent and received via IP address instead of e-mail account.

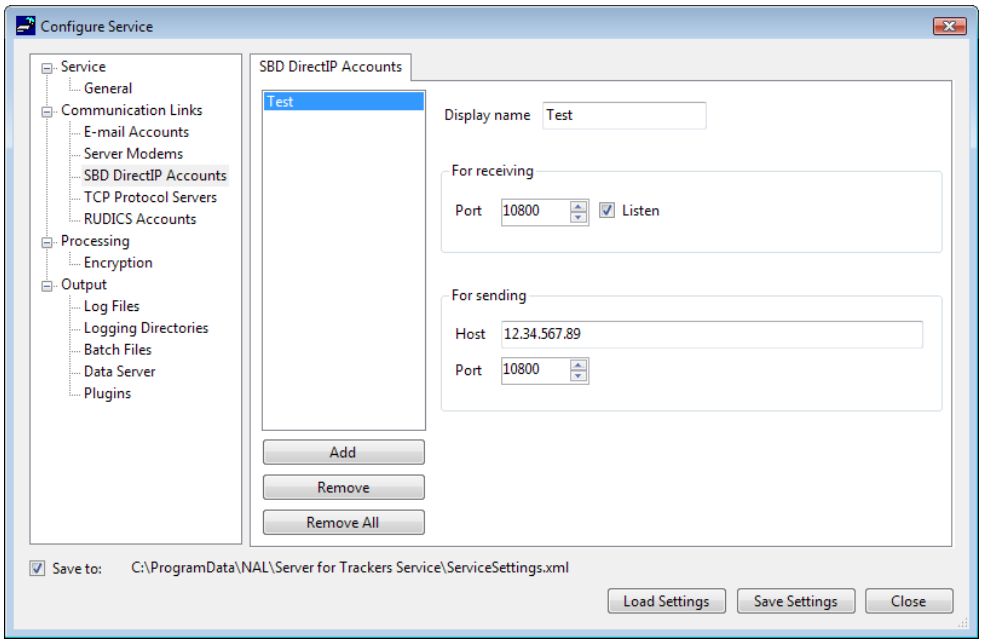


Figure 8. Configure Service – SBD DirectIP Accounts Screen.

In Server for Trackers, the SBD DirectIP account communication link allows the user to take advantage of SBD DirectIP. There are setting for receiving as well as sending. For receiving, the port setting is the port that the gateway will connect to when delivering MO messages. For sending, the host and port settings describe the end point that will be used to send MT messages. In order for the SBD DirectIP accout to accept connections the listen setting must be selected.

Note that there may be additional setup needed to begin receiving SBD messages. For instance the computer that hosts the service should be configured with the IP address that the user's SBD DirectIP account was setup for. Also, if there is a firewall or router, it should be configured to allow the SBD DirectIP gateway through.

When the gateway connects to an SBD DirectIP account to deliver an SBD message, another communication link called an "SBD DirectIP Client Handler" will appear on the main screen in the list of communication links. This is a temporary communication link which the SBD DirectIP accout spawns to handle the connection since there may be mulitple connections at the same time. This communication linik will disappear once the connection has ended.

Just as a temporary communication link is created when receiving a MO message, one is also created when sending a MT message. However, the link created when sending a MT message is called "SBD DirectIP Client" and will not disappear once the MT is sent. Instead, the link is kept around so that the user can see whether or not the send attempt was successful. Once the status has been reviewed, the link may be removed by selecting it on the main screen and sending the "Remove" command to the service.

TCP Protocol Servers

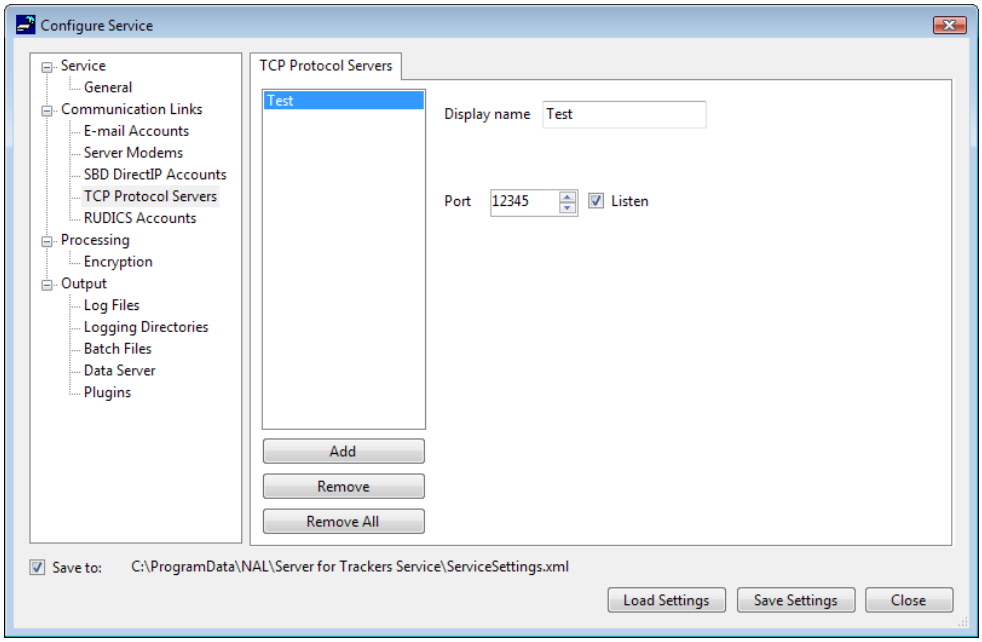


Figure 9. Configure Service – TCP Protocol Servers Screen.

TCP protocol servers allow Server for Trackers to communicate with remote modems over a GSM network using TCP. In order for a TCP protocol server to receive connections, a port must be specified and

the listen setting must be selected. When a connection is received, the TCP protocol server creates a temporary communication link called a "TCP Protocol Client Handler" which will handle the connection so that there may be multiple connection at the same time. The "TCP Protocol Client Handler" communication link will appear on the main screen in the list of communication links and then disappear once the connection has ended.

RUDICS Accounts

A NOC can communicate with remote satellite modems using RUDICS. RUDICS allows the NOC to receive or initiate a data call with a remote modem through a Telnet connection. This is possible because the gateway maintains a bank of modems which handle the satellite connection with the remote modem. The gateway just passes data from the remote modem to the Telnet connection and vice versa. This setup is illustrated below.

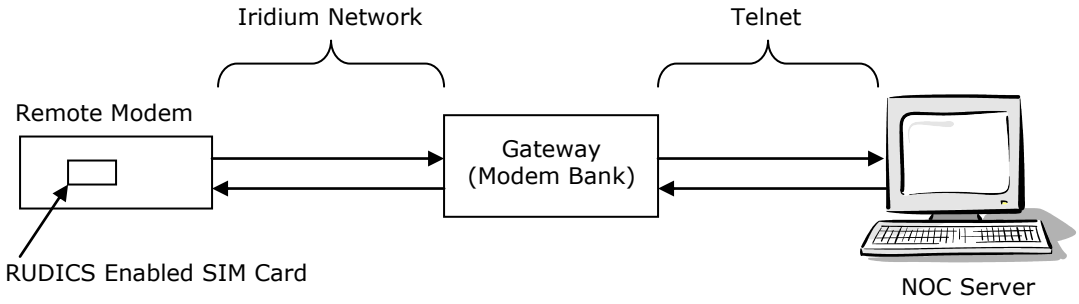


Figure 10. RUDICS Operational Diagram.

Prior to setting up a RUDICS account by an Iridium Service Provider, the following information must be given by the account holder: SIM cards to be enabled for that particular RUDICS account and the NOC IP address port. Once a RUDICS account has been setup, the following information is given to the account holder: DNIS (Group #) and the RUDICS gateway IP address and ports. When a remote modem places a RUDICS call to the NOC, it must have a RUDICS enabled SIM card installed and must dial the DNIS number. The RUDICS system at the Iridium gateway then uses the NOC's IP address and port to initiate a socket connection with the NOC. When the NOC places a RUDICS call to a remote modem, the NOC must first establish a socket connection with the RUDICS system using the RUDICS gateway IP address and one of the RUDICS gateway ports. After connecting to the RUDICS gateway, the NOC can dial the MSISDN-C number of remote modem to connect to.

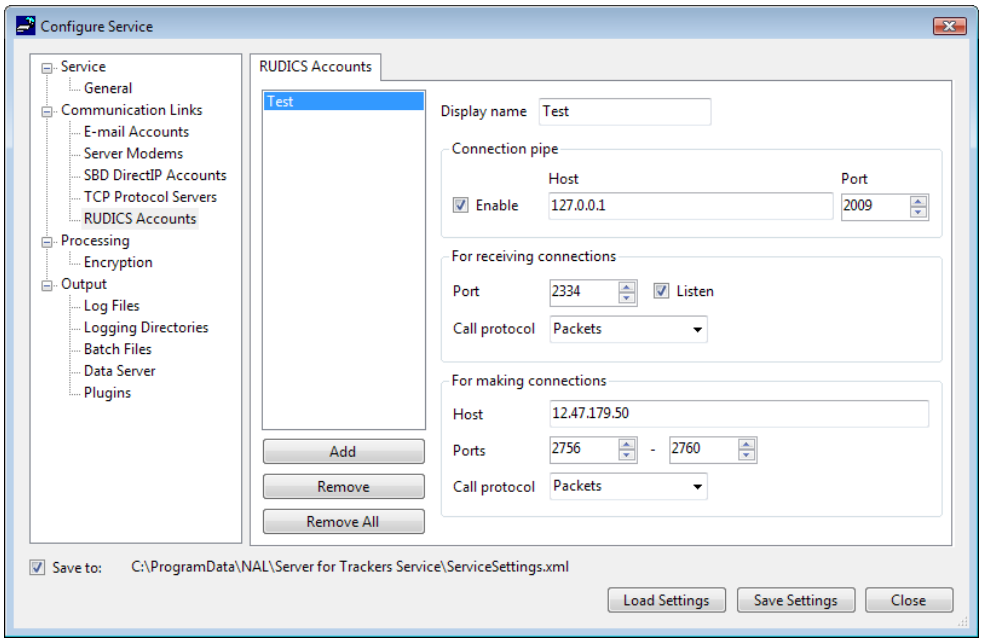


Figure 11. Configure Service – RUDICS Accounts Screen.

On the "RUDICS Accounts" screen, the NOC port should be entered in the "For receiving connections" group box and the RUDICS gateway IP address and ports should be entered in the "For making connections" group box. Also, if there is a firewall on the NOC, it should be configured to allow RUDICS connections.

The settings in the "Connection pipe" group box are for causing the RUDICS account to open a socket connection whenever a RUDICS connection is made. Received data will be sent to this pipe, and data received from this pipe will be sent to the remote modem. If encryption is enabled, data received from the pipe will be encrypted before being sent to the remote modem and data received from the remote modem will be decrypted before being sent to the pipe. Likewise, if the call protocol is "Packets", Server for Trackers handles the packets and only passes the data stream to the pipe.

The "Call protocol" settings determine how data is sent and received from the remote modem. There are two options for the call protocol. The "None" option, means that no protocol will be used and that data will be sent and received as is. The "Packets" protocol means that data will be sent and received in packets. By using packets, data transmission is more reliable allowing even large files to be transferred. The call protocol can be set independently for received connections and initiated connections.

When the gateway connects to a RUDICS account, another communication link called a "RUDICS Client Handler" will appear on the main screen in the list of communication links. This is a temporary communication link which the RUDICS account spawns to handle the connection since there may be mulitple connections at the same time. This communication linik will disappear once the connection is over.

Another temporary communication link called a "RUDICS Client" is created when the "Connect..." command is executed for a RUDICS account. This link will also disappear once the connection is over or if a connection could not be established.

4.3 Processing

Encryption

Server for Trackers is able to encrypt and decrypt data. If the "Use encryption" setting is set, Server for Trackers will attempt to encrypt outgoing data and decrypt incoming data. However, it will fail if the encryption user is not logged in or if there is not a key associated with the encryption user for the remote modem. If Server for Trackers fails to decrypt incoming data, the data will be discarded. Therefore, it is important to make sure that the encryption settings are correct.

For decrypting SMS messages the keys should be assoicated with the phone number of the remote modem without the "00" international prefix since Server for Trackers looks up the keys for decrypting SMS by phone number. For SBD, the keys should by stored by IMEI. For RUDICS calls, the keys can be stored either by phone number or IMEI.

To enter and modify users and keys, click the "Open Crypto Officer..." button. This will bring up the "Crypto Officer" window. Details regarding the "Crypto Officer" window are intentionally left out in this section. Users can read the Encryption Module Manual TN2007-636-V2.1.0 for more information.

When changes are saved in the "Crypto Officer" window, they will be saved to the Crypto Officer files without the need to click the "Save Settings" button in the "Configure Service" window. The changes that are made using the "Crypto Officer" window will only be available to the encryption user the next time the encryption user logs in. Therefore, if the encryption user was logged in while the changes were made, then the encryption user should be logged out and then logged back in. This will require unchecking "Log in for encryption", saving the settings, checking "Log in for encryption", and saving the settings again.

The encryption user login information is stored in the settings file in an encrypted format for security. This allows the service to log the encryption user in automatically when it starts up.

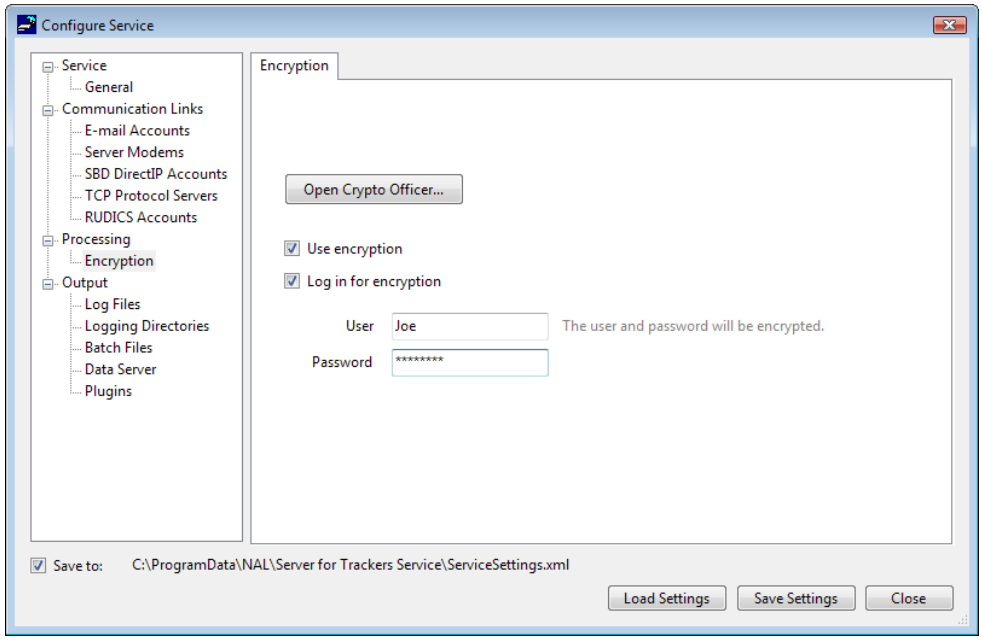


Figure 12. Configure Service – Encryption Screen.

4.4 Output

Server for Trackers has five ways to output the data it receives – log files, logging directories, batch files, a data server, and plugins. These output methods will be described in the following sections.

For each output method that is setup, the user can choose what protocols and data types to process. Also, for each data type, the user can choose how the data should be formatted before being processed by the corresponding output method.

Formatting settings, including a format and optional XSL stylesheet, can be applied to specific data types, such as "NAL GPS Report 5", or to categories, such as "GPS Reports". Since specific data types have priority over categories, if a specific type is setup as well as the corresponding category, then data belonging to the specific type will be formatted according to the settings for the specific type rather than the settings for the category.

The "XML" format will cause the data to be formatted in a standard XML format according to its type. For details on the standard XML formats, see Appendix A. The "Hex" format will cause the data to be formatted as a hex string inside of an XML element named "hex". The "Binary" format will cause the data to not be formatted at all. Since the "Binary" format outputs raw data, it is only available for logging directories. Both the "XML" and "Hex" formatted data will be wrapped in an XML element containing metadata. The metadata wrapper XML element is as follows.

```
<data>
    <meta>
        <sender type=""></sender>
        <receiver type=""></receiver>
        <time></time>
        <protocol></protocol>
        <type></type>
        <sessionStart></sessionStart>
    </meta>
    <!-- Formatted data goes here -->
</data>
```

TAG NAME	DESCRIPTION
sender	The remote modem that sent the data. The type attribute can be "Identifier", "Phone Number", or "IMEI". If the sender is unknown, this element will not be present.
receiver	The communication link that received the data. The possible values for the type attributte along with the corresponding element value description are shown below. "E-mail Account" Display name "Server Modem" Port name "SBD DirectIP Client Handler" Remote end point (IP address and port) "TCP Protocol Client Handler" Remote end point (IP address and port) "RUDICS Client" Remote end point (IP address and port)

TAG NAME	DESCRIPTION
	"RUDICS Client Handler" Remote end point (IP address and port)
time	The UTC time the the data was received.
protocol	The protocol that was used to send the data. Can be "SBD", "SMS", "Call", or "TCP".
type	The type of the data. Can be one of the following. "Other" "NAL GPS Report 3" "NAL GPS Report 4" "NAL GPS Report 5" "NAL GPS Report 6" "NAL 10 Byte GPS Report 0" "PECOS P3 GPS Report" "PECOS P4 GPS Report" "Update Response 0" "Update Response 1" "Update Response 2" "Update Response 3" "Status Report 0"
sessionStart	The UTC time that the call began. Only present if the protocol is "Call" or "TCP".

After the data is formatted according to the format setting, it will be transformed using the XSL file specified by the stylesheet setting. The "Binary" format does not allow the user to specify a stylesheet since the data is not formatted into XML like the other formats. Several XSL stylesheets are installed with Server for Trackers in order to replicate the old builtin formats. These XSL file are located in the "NAL\Server for Trackers Service\XSL" subfolder of the common application data folder. Only the files that do not have "Helper" in the name should be chosen when setting the data type settings. Also, these XSL files only work for GPS reports and therefore should only be associated with GPS reports.

NOTE: If processing fails, the data will be discarded before being sent the the output channels. Processing can fail either because SMS data is not in the correct format or decryption failed. Also, the parts of a multipart SMS message are buffered and only sent to the output channels once all of the parts have been received.

Log Files

A log file is a file which Server for Trackers will append data to.

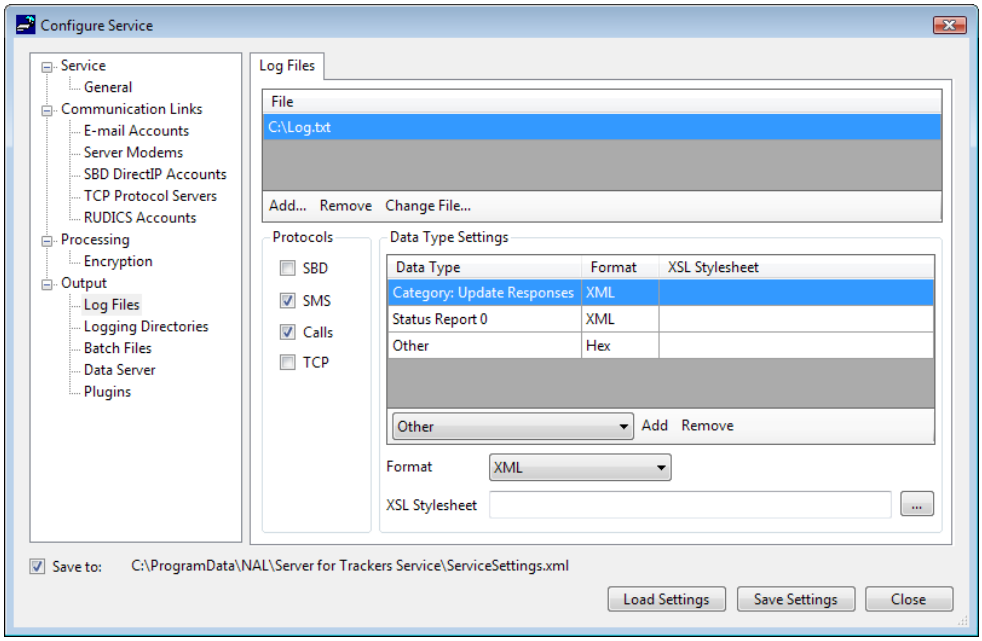


Figure 13. Configure Service – Log Files Screen.

Logging Directories

A logging directory is a directory which Server for Trackers will log data to. Each piece of data will be saved to a unique file in the directory. The file name will be determined by a user defined file prefix, a UTC time stamp, and an optional counter as follows...

{File Prefix}{YYYY}_{MM}_{DD}_{HH}_{mm}_{SS}_{FFF}[_{COUNTER}].xml

The file prefix is a string specified by the user to distinguish the files from other files with similar names. The {HH} is the hour according to a 24 hour clock. {FFF} is the fractions of a second. Elements of the date and time will be padded on the left with zeros so that they are always the same width. If there is already a file with the same name in the logging directory, an underscore followed by a counter is appended to the file name before the extension. This counter starts at 1 and increments by 1 until there is no longer a conflict.

Example:
File Prefix: Test_
Date and Time: April 23, 2009 3:47:52.168 PM
File Name: Test_2009_04_23_15_47_52_168.xml
File Name (with counter): Test_2009_04_23_15_47_52_168_1.xml

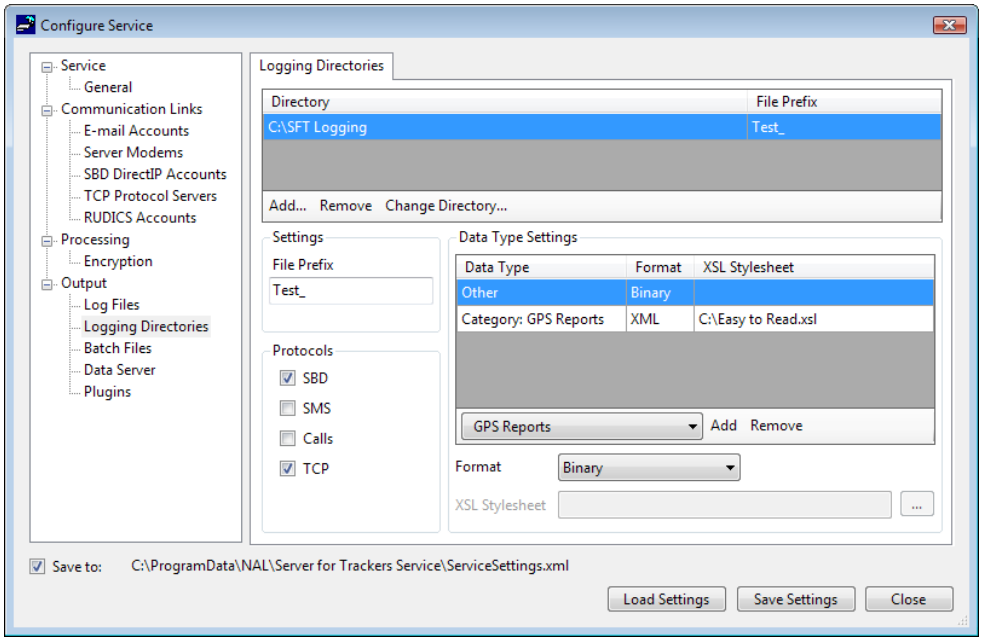


Figure 14. Configure Service – Logging Directories Screen.

Batch Files

A batch file is a file that contains a batch program or a user-defined program that Server for Trackers executes when data is received. The batch file may be either a command line batch file (.bat) or a Windows executable file (.exe). The batch file is executed once for each data item received. When the batch file executes, the data is passed as a parameter to the batch file.

The batch file could be an FTP program sending the current GPS report to multiple servers via FTP each time a GPS report is received by the NOC server computer. Or perhaps it could be a mapping program reading, manipulating and displaying real-time incoming GPS reports.

The first argument passed to the batch file will always be the path to the batch file. The second argument will be the data.

NOTE: The .bat batch files need to be named such that they have the same name with the command line as with Windows or the batch file will not be executed. If a batch file is not executing properly, try applying the following naming conventions to the batch file and folder where it is located: The file name should be no more than 8 characters with no spaces and a 3 character extension. (i.e. BATCH001.BAT) The folder name and all parent folders should be no more than 8 characters with no spaces. Also, note that a Windows executable is the method of choice for most using the "Batch File" functionality.

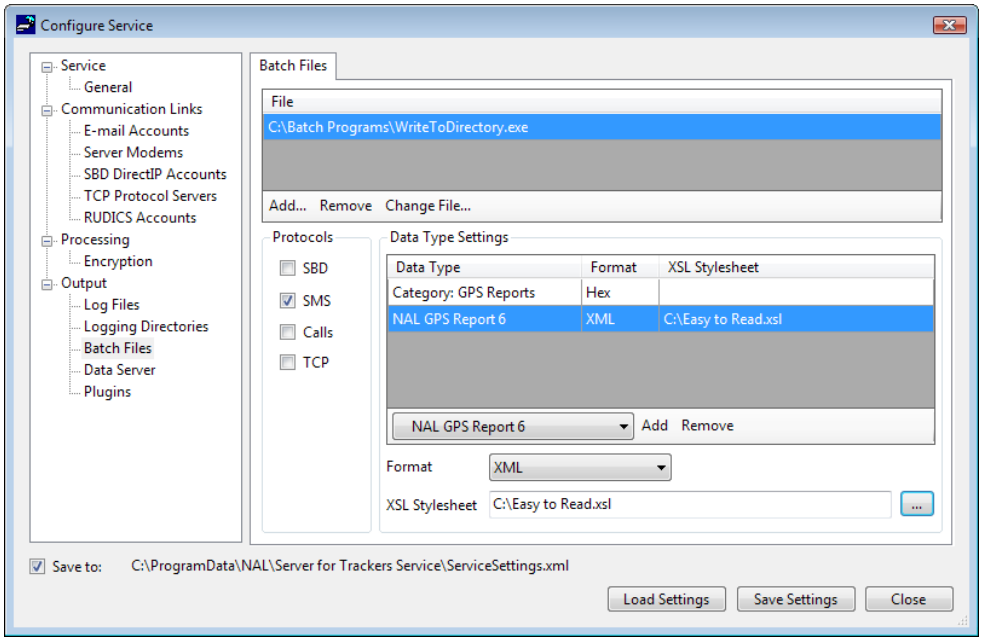


Figure 15. Configure Service – Batch Files Screen.

Data Server

The data server is a server to which user programs may connect to in order to receive data from Server for Trackers. Multiple programs may be connected to the server at the same time. Since the data server is a server, a free port needs to be chosen. To allow programs to connect, check the "Listen" checkbox (note that settings do not go into effect until they are saved). To disconnect all connected programs and to disallow programs from connecting, uncheck the "Listen" checkbox.

The Server for Trackers Controller has the ability to connect to the data server. To connect the controller to the data server, click the data server "Connect" button at the bottom of the main window. Of course, the controller will need to be setup to connect to the correct port by using the "Configure Controller" window. After the controller is connected, whenever it receives the formatted output via the data server, the formatted output will be displayed on the left side of the main window.

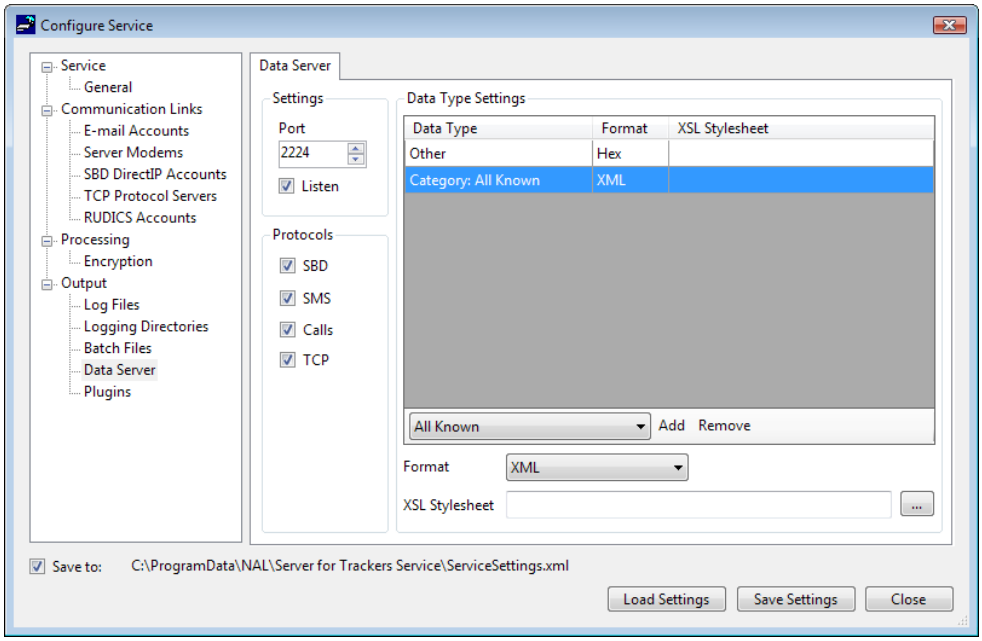


Figure 16. Configure Service – Data Server Screen.

Plugins

A plugin is a .NET DLL that contains a method that Server for Trackers executes when data is received. The plugin must contain a class which implements the IProcessDataPlugin interface and which has the ProcessDataPluginAttribute attribute defined. The IProcessDataPlugin interface requires a method with the signature - void ProcessData(string). Both the IProcessDataPlugin interface and the ProcessDataPluginAttribute attribute can be found in Nal.ServerForTrackers.Plugins.dll, which is in the installation directory of Server for Trackers Service.

The implementor of the plugin should be aware that the plugin will be loaded into the Server for Trackers Service memory and run as though it were part of the service (unlike a batch file which is spawned and has its own memory space and thread). Removing the plugin will prevent it from being called. However, the plugin's dll will still be loaded into memory. Therefore, after removing a plugin, the service should be restarted.

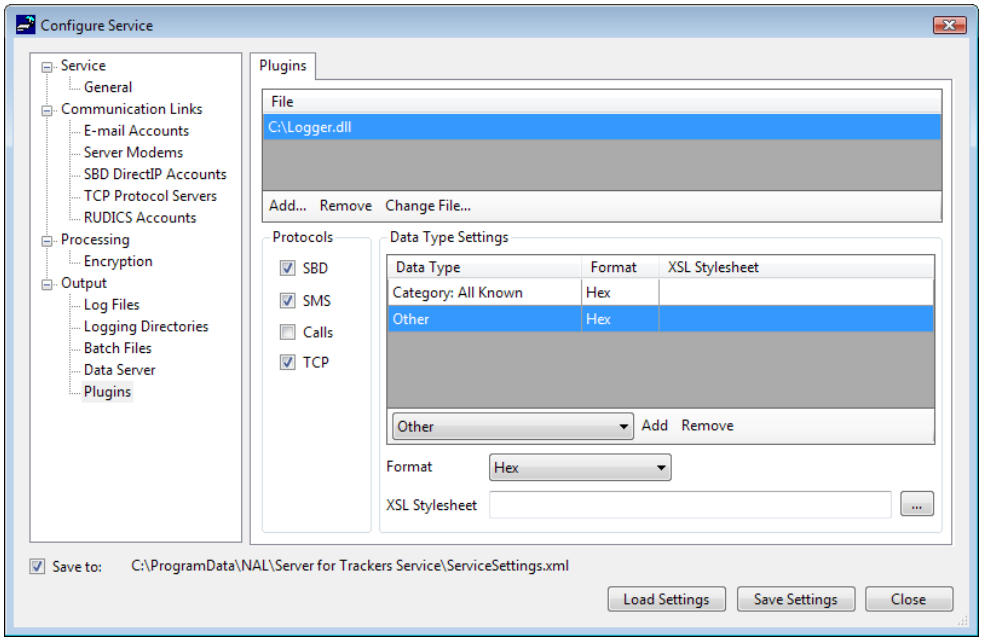
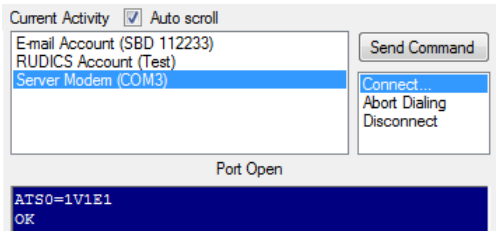


Figure 17. Configure Service – Plugins Screen.

5.0 CONTROLLING THE SERVICE



When a communication link is selected on the main screen, a list of possible commands is shown to the right. To execute a command for the selected communication link, select the command and then click the "Send Command" button. The communication link will attempt to execute the command. For some commands, however, another window will pop up. This is because the command needs some parameters. If the command will pop up another window is it appended with "...". One example of this is the "Connect..." command for server modems. This command will bring up the "Contacting Remote Modems" window with the appropriate communication link already selected. If a command could not be executed, the service will send an error message back to the controller, which the controller will display. If the service is able to execute a command, typically the status and activity located just below the list of communication links will indicate the progress of the command.

5.1 Contacting Remote Modems

The "Contacting Remote Modems" window is used to contact a remote modem. This window can be opened from the "Options | Contact Remote Modem..." menu item. To contact a remote modem, chose a communication link, a protocol, and a number (either from the number list or enter one). Then click the "Contact" button.

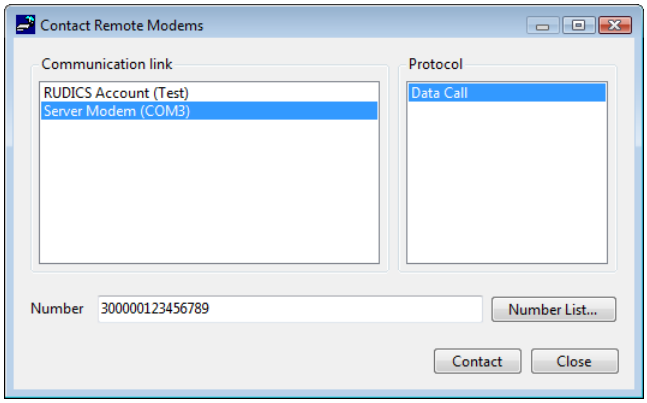


Figure 18. Contact Remote Modems Screen.

NOTE: Only data calls are supported in this version. Therefore, only RUDICS accounts and server modems will appear in the communication links list box and only "Data Call" will appear in the protocol list box.

NOTE: A SIM card inside a remote modem has two numbers: MSISDN and MSISDN-C if the card supports both voice and data. The MSISDN number must be dialed if calling the remote modem with an Iridium modem. The MSISDN-C number, along with the appropriate prefix, must be dialed if calling the remote modem with a landline modem. To avoid high international airtime costs, an Iridium satellite modem should be used by the NOC server computer when calling remote modems instead

of a landline modem. If the RUDICS account is used to connect to the remote modem, then the MSISDN-C number should be used. If the card is only provisioned for data then the MSISDN-C number is used for both Iridium to Iridium calls and for calling the Iridium from a landline.

6.0 MONITORING THE SERVICE

Since the core functionality of Server for Trackers is in the service, it is important to be able to monitor the service. The "Monitor Service Status" window, which can be opened from the "Options | Monitor Service Status..." menu item, allows the user to monitor some key aspects of the service.

6.1 Messages

The "Messages" screen in the "Monitor Service Status" window is used by the controller to display messages received from the service while the controller is connected to the service. However, the "Query Current Service Status" button on this screen can be clicked at any time to request a status update from the service. The service will then send back a report of critical status information, such as whether servers are listening, whether com ports are open, and whether the encryption user is logged in.

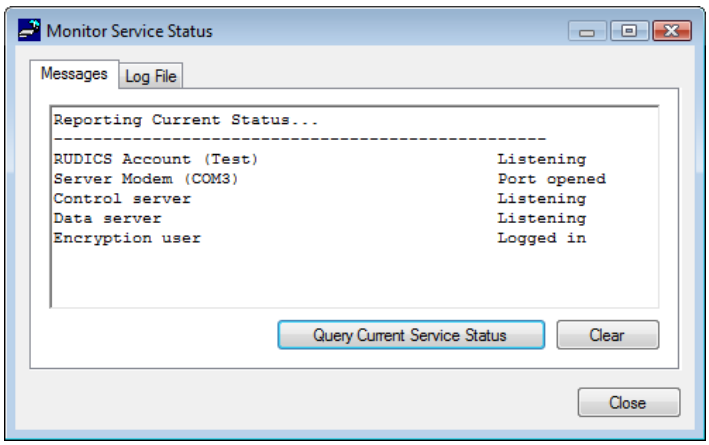


Figure 19. Monitor Service Status – Messages Screen.

6.2 Log File

Although the "Messages" screen keeps the user updated while the controller is connected to the service, the controller may not always be connected to the service. Therefore, the service logs crucial information to a log file, such as changes in settings, starting and stopping, and warnings when emergency reports are received. The service log file is named "ServiceLog.txt" and is located in the "NAL\Server for Trackers Service" subfolder of the common application data folder. To view the log file from the controller, the "Log File" screen in the "Monitor Service Status" window may be used. The screen will reload the log file whenever the "Refresh" button is clicked.

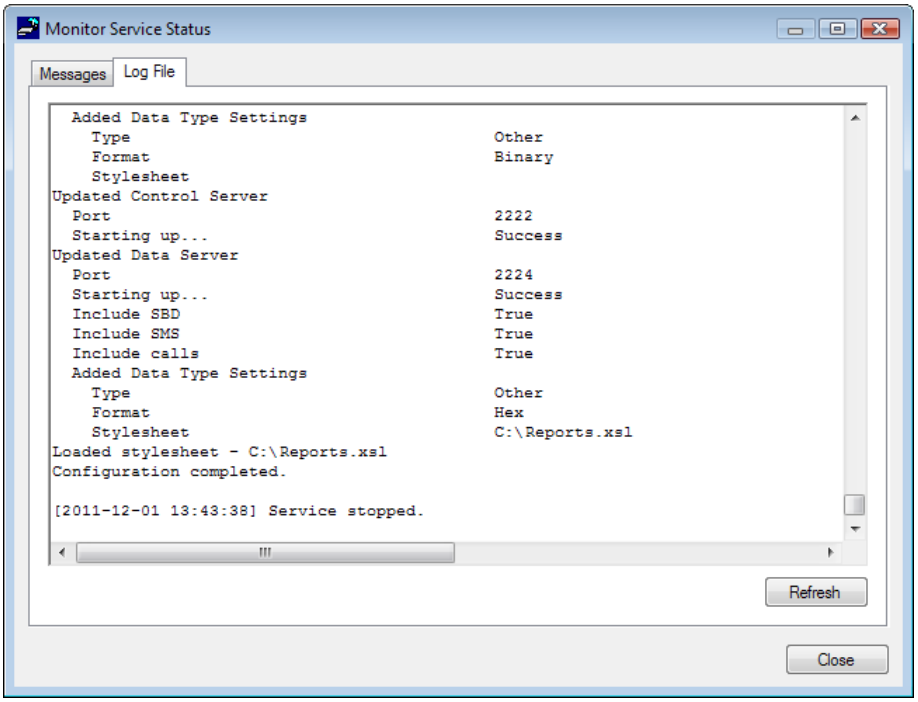


Figure 20. Monitor Service Status – Log File Screen.

7.0 SENDING REMOTE UPDATES

Updates can be sent to remote modems by using the Send Remote Updates screen. This screen can be accessed by choosing "Send Remote Updates…" from the "Options" menu.

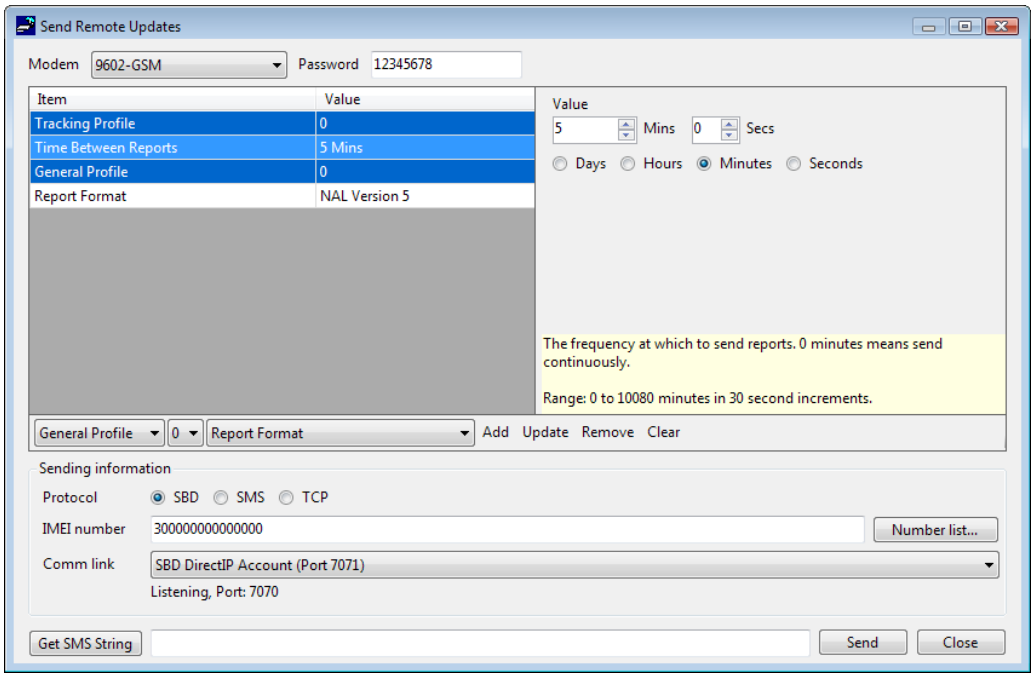


Figure 21. Send Remote Updates Screen.

To send a remote update, first choose the type of modem that will receive the remote update using the "Modem" combo box. Below the "Modem" combo box there is an area where the remote update can be edited. Settings included in the remote update will be shown on the left while an interface for editing the selected setting will be shown on the right. At the bottom of this area, just above the "Sending information" group box are combo boxes with all of the possible settings that can be updated for the selected modem.

To add a setting to the remote update, select it in the combo boxes and then click the "Add" button. To edit a setting, select one of the already added settings in the list and use the interface to change the value. Once the value has been modified, click the "Update" button to update the setting with the new value. Settings can be removed by selecting them and clicking the "Remove" button. To clear the remote update, use the "Clear" button.

Once the remote update has been created, use the "Sending information" group box to specify how to send the remote update and to which modem to send it. Depending on the type of modem that the remote update is being sent to, the remote update may be sent by SBD, SMS, or TCP. If the SMS protocol is selected, the remote modem needs to be specified by its phone number, otherwise, it can specified by its IMEI number. The number list, which can be setup by using the "Configure Controller…" screen, can be used to easily load an IMEI or phone number by clicking the "Number list…" button. The communication link combo box will be filtered according to the selected protocol. For SMS an email account must be chosen and for SBD either an email account or an SBD DirectIP account must be chosen. However, for TCP there is no

need to select a communication link since Server for Trackers cannot initiate a TCP connection to the remote modem. Instead, when the "Send" button is clicked, Server for Trackers will place the remote update in a queue and wait for the next TCP server connection that reports a matching IMEI number. Queued TCP updates are stored in a file called "TcpServerQueues.xml" in the "NAL\Server for Trackers Service" subfolder of the common application data folder. This file is only read once at startup and is updated whenever a remote update is queued or successfully delivered.

After the remote update has been created and the sending information has been specified, click the "Send" button to send the remote update to the remote modem. The "Get SMS String" button gets a string that can be sent manually using an external SMTP client.

The method of determining whether the send was successful or not, depends on the communication link being used. For email accounts, the SMTP portion of the status below the "Comm link" combo box will go from "Disconnected" to "Sending Message" and then back to "Disconnected". If there was an error while sending, it will be shown in the status after the "Disconnected" text. For determining the result of sending using an SBD DirectIP account, the status shown below the "Comm link" combo box is not useful since the SBD DirectIP account creates another communication link called a SBD DirectIP client to do the sending. Therefore, the status of the SBD DirectIP client will need to be checked to see the result of the send operation. To see the status of the SBD DirectIP client, go to the main screen and select the SBD DirectIP client in the list of communication links. The status should show a confirmation status. Below is a table of possible confirmation statuses and their meanings.

STATUS	DESCRIPTION
1 – 50	Successful, order of message in the MT message queue
0	Successful, no payload in message
-1	Invalid IMEI – too few characters, non-numeric characters
-2	Unknown IMEI – not provisioned on the GSS
-3	Payload size exceeded maximum allowed
-4	Payload expected, but none received
-5	MT message queue full (max of 50)
-6	MT resources unavailable
-7	Violation of MT DirectIP protocol
-8	Ring alerts to the given IMEI are disabled
-9	The given IMEI is not attached (not set to receive ring alerts)

Once the SBD DirectIP client has finished sending, it is no longer used by the program and therefore should be removed by selecting the "Remove" command and then clicking the "Send Command" button on the main screen. Regardless, of the communication link used to transmit the remote update, if the remote update was successfully sent, it will appear on the "Track Remote Updates" screen covered in the next section.

8.0 TRACKING REMOTE UPDATES

Server for Trackers tracks remote updates by storing remote update requests and responses in a file named "RemoteUpdatePairs.xml" in the "NAL\Server for Trackers Service" subfolder of the common application data folder. This file is made up of "Pair" elements with "Request" and "Response" subelements as shown below.

<Pair>
<Request To="I:300000000000000" Sent="20101202221847">{Base64 Request Data}</Request>
<Response From="P:881234567890" Received="20101202231112">{Base64 Response Data}</Response>
</Pair>

The "To" and "From" attributes contain an IMEI or phone number, which is indicated by a prefix of "I:" for IMEI or "P:" for phone number. The "Sent" and "Received" attributes contain the UTC date with the four digit year coming first followed in order by month, day, hour, minute, and second all of which are two digits. The contents of the "Request" and "Response" elements are the raw request or response data in Base64 format.

Whenever a remote update request is sent from Server for Trackers, a new "Pair" element with only the "Request" subelement is added to the file. Then, whenever a remote update response is received by Server for Trackers, Server for Trackers will try to match the response to a request in the file. If a match is found, then the response is logged in the same "Pair" element with the request that it was matched to. If the response does not match any of the requests, then it is logged in a new "Pair" element without the "Request" subelement. Server for Trackers will only look at the unmatched requests when trying to make a match.

The "Track Remote Updates" screen which can be opened by choosing "Track Remote Updates…" from the "Options" menu can be used to view the contents of the "RemoteUpdatePairs.xml" file in a user friendly way. To refresh the screen with the updated contents of the file, click the "Refresh" button. The "Show local time" check box can be used to display the times in your computer's time zone instead of UTC.

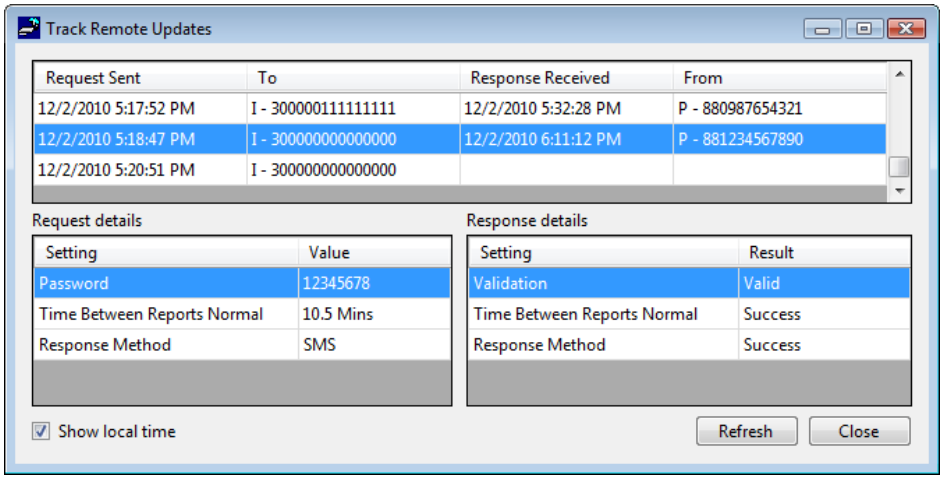


Figure 22. Track Remote Updates Screen.

APPENDIX A – XML FORMATS

This section contains descriptions of the standard XML ouput for the various types of data that Server for Trackers recognizes. In the descriptions, [] denotes how many times an element may appear in its context. If there is not [] for an element, then that element must appear exactly once in its context. {} denotes the type of data. The following table describes the types.

TYPE	DESCRIPTION
Integer	An integer. May or may not have a negative sign.
Float	A floating point number. May or may not have a negative sign or decimal point.
Boolean	Either 1 meaning true or 0 meaning false
DateTime	UTC date time represented according to the XML Schema 1.1 dateTimeStamp data type
HexString	A string consisting of hex characters, where every two characters represents a byte
Enum	A defined set of values. These values will be described in the table of element descriptions which follows the XML description.

GPS Reports

NAL GPS Report 3

```
<nalGpsReport3>
    <id>{String}</id> [0..1]
    <point> [1..*]
        <time>{DateTime}</time>
        <lat>{Float}</lat>
        <lng>{Float}</lng>
        <alt>{Float}</alt>
        <gndVel>{Float}</gndVel>
        <course>{Float}</course>
        <verVel>{Float}</verVel>
        <fix>{Enum}</fix>
        <sats>{Integer}</sats>
        <emer>{Boolean}</emer>
    </point>
</nalGpsReport3>
```

NAL GPS Report 4

```
<nalGpsReport4>
    <id>{String}</id> [0..1]
    <point> [1..*]
        <time>{DateTime}</time>
        <lat>{Float}</lat>
        <lng>{Float}</lng>
        <alt>{Float}</alt>
        <gndVel>{Float}</gndVel>
        <course>{Float}</course>
        <verVel>{Float}</verVel>
        <fix>{Enum}</fix>
        <sats>{Integer}</sats>
        <hdop>{Float}</hdop>
        <vdop>{Float}</vdop>
        <motion>{Boolean}</motion>
        <emer>{Boolean}</emer>
        <emerAcked>{Boolean}</emerAcked>
    </point>
</nalGpsReport4>
```

NAL GPS Report 5

```
<nalGpsReport5>
    <time>{DateTime}</time>
    <lat>{Float}</lat>
    <lng>{Float}</lng>
    <alt>{Float}</alt>
    <gndVel>{Float}</gndVel>
    <course>{Float}</course>
    <verVel>{Float}</verVel>
    <fix>{Enum}</fix>
    <sats>{Integer}</sats>
    <hdop>{Float}</hdop>
    <vdop>{Float}</vdop>
    <motion>{Boolean}</motion>
    <emer>{Boolean}</emer>
    <emerAcked>{Boolean}</emerAcked>
    <inputPins>{Integer}</inputPins>
    <outputPins>{Integer}</outputPins>
    <id>{String}</id> [0..1]
</nalGpsReport5>
```

NAL GPS Report 6

```
<nalGpsReport6>
    <time>{DateTime}</time>
    <lat>{Float}</lat>
    <lng>{Float}</lng>
    <alt>{Float}</alt>
    <gndVel>{Float}</gndVel>
    <course>{Float}</course>
    <verVel>{Float}</verVel>
    <fix>{Enum}</fix>
    <sats>{Integer}</sats>
    <hdop>{Float}</hdop>
    <vdop>{Float}</vdop>
    <motion>{Boolean}</motion>
    <emer>{Boolean}</emer>
    <emerAcked>{Boolean}</emerAcked>
    <abCode>{Integer}</abCode>
    <cmCode>{Integer}</cmCode>
    <freeText>{String}</freeText> [0..1]
    <routing> [0..1]
        <email>{String}</email> [0..*]
        <phone>{String}</phone> [0..*]
        <imei>{String}</imei> [0..*]
    </routing>
</nalGpsReport6>
```

NAL 10 Byte GPS Report 0

```
<nal10ByteGpsReport0>
    <time>{DateTime}</time>
    <lat>{Float}</lat>
    <lng>{Float}</lng>
    <fix>{Enum}</fix>
    <pdop>{Float}</pdop>
    <motion>{Boolean}</motion>
    <emer>{Boolean}</emer>
    <emerAcked>{Boolean}</emerAcked>
</nal10ByteGpsReport0>
```

PECOS P3 GPS Report

```
<pecosP3GpsReport>
    <imei>{String}</imei>
    <time>{DateTime}</time>
    <brevity>{Integer}</brevity>
    <pdop>{Float}</pdop>
    <lat>{Float}</lat>
    <lng>{Float}</lng>
    <alt>{Float}</alt>
    <course>{Float}</course>
    <gndVel>{Float}</gndVel>
</pecosP3GpsReport>
```

PECOS P4 GPS Report

```
<pecosP4GpsReport>
    <imei>{String}</imei>
    <time>{DateTime}</time>
    <brevity>{Integer}</brevity>
    <pdop>{Float}</pdop>
    <lat>{Float}</lat>
    <lng>{Float}</lng>
    <alt>{Float}</alt>
    <course>{Float}</course>
    <gndVel>{Float}</gndVel>
    <fix>{Enum}</fix>
    <hdop>{Float}</hdop>
    <vdop>{Float}</vdop>
</pecosP4GpsReport>
```

ELEMENT	DESCRIPTION
abCode	Address book code
alt	Height above mean sea level in meters
brevity	Brevity code
cmCode	Canned message code
course	Course in degrees from True North
emer	Emergency indicator
emerAcked	Whether the modem has received an emergency acknowledgment for its current emergency state
fix	Values: Other The fix is something other than the values supported by the report format DR Dead reckoning was used (Supported by versions NAL 3, 4, and 5) 2D The fix is a 2D fix (Not supported by the NAL 10 Byte 0) 3D The fix is a 3D fix (Only supported by PECOS P4) Valid 3D The fix is a 3D fix and is valid according to the range parameters (Not supported by PECOS P4)
freeText	Free-text message
gndVel	Ground velocity in kilometers per hour
hdop	Horizontal delusion of precision
id	The configured identifier of the modem.
imei	The IMEI number of the modem.
inputPins	Each bit represents the level of an input pin. Bit 0 is the corresponds to pin 0, bit 1 corresponds to pin 1, and so forth.
lat	Latitude
lng	Longitude
motion	Whether the report was sent because the modem was awakened by the motion sensor
outputPins	Each bit represents the level of an output pin. Bit 0 is the corresponds to pin 0, bit 1 corresponds to pin 1, and so forth.
pdop	Position dilution of precision
sats	Number of GPS satellites used to determine the position
time	The UTC time when the GPS information was gathered.
vdop	Vertical delusion of precision
verVel	The vertical component of the velocity in meters per second. Ascending is negative and descending is positive.

Update Responses

Update Response 0

```
<updateResponse0>
    <status>{Enum}</status>
    <request>
        <time>{DateTime}</time>
        <timings> [0..1]
            <timeBetweenReports>{Integer}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </timings>
        <callable>{Boolean}</callable> [0..1]
        <deliveryShortCode>{Integer}</deliveryShortCode> [0..1]
    </request>
</updateResponse0>
```

Update Response 1

```
<updateResponse1>
    <status>{Enum}</status>
    <inputPinStates>{Integer}</inputPinStates>
    <request>
        <time>{DateTime}</time>
        <normalTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </normalTimings>
        <testTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </testTimings>
        <emergencyTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </emergencyTimings>
        <normalCallable>{Enum}</normalCallable> [0..1]
        <testCallable>{Enum}</testCallable> [0..1]
        <emergencyCallable>{Enum}</emergencyCallable> [0..1]
        <motionSensorAwakes> [0..1]
            <inNormalMode>{Boolean}</inNormalMode>
            <inTestMode>{Boolean}</inTestMode>
            <inEmergencyMode>{Boolean}</inEmergencyMode>
        </motionSensorAwakes> [0..1]
        <deliveryShortCode>{Integer}</deliveryShortCode> [0..1]
```

```
        <blockInvalidGpsReports>{Enum}</blockInvalidGpsReports> [0..1]
        <emergencyReportFlood>{Integer}</emergencyReportFlood> [0..1]
        <outputPinStates>{Integer}</outputPinStates> [0..1]
    </request>
</updateResponse1>
```

Update Response 2

```
<updateResponse2>
    <status>{Enum}</status>
    <inputPinStates>{Integer}</inputPinStates>
    <request>
        <time>{DateTime}</time>
        <normalTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </normalTimings>
        <testTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </testTimings>
        <emergencyTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </emergencyTimings>
        <normalAwakeTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </normalAwakeTimings>
        <testAwakeTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </testAwakeTimings>
        <emergencyAwakeTimings> [0..1]
            <timeBetweenReports>{Float}</timeBetweenReports>
            <timeToKeepTrying>{Integer}</timeToKeepTrying>
        </emergencyAwakeTimings>
        <normalCallable>{Enum}</normalCallable> [0..1]
        <testCallable>{Enum}</testCallable> [0..1]
        <emergencyCallable>{Enum}</emergencyCallable> [0..1]
        <motionSensorAwakes> [0..1]
            <inNormalMode>{Boolean}</inNormalMode>
            <inTestMode>{Boolean}</inTestMode>
```

```
            <inEmergencyMode>{Boolean}</inEmergencyMode>
        </motionSensorAwakes>
        <motionSensorBegin> [0..1]
            <windowCount>{Integer}</windowCount>
            <sensitivity>{Integer}</sensitivity>
        </motionSensorBegin>
        <motionSensorEnd>{Integer}</motionSensorEnd> [0..1]
        <normalMotionSensorWait>{Integer}</normalMotionSensorWait> [0..1]
        <testMotionSensorWait>{Integer}</testMotionSensorWait> [0..1]
        <emergencyMotionSensorWait>{Integer}</emergencyMotionSensorWait> [0..1]
        <normalSamePlaceSkipReports> [0..1]
            <mode>{Enum}</mode>
            <radius>{Integer}</radius>
            <cyclesBeforeSkipping>{Integer}</cyclesBeforeSkipping>
            <cyclesToSkip>{Integer}</cyclesToSkip>
        </normalSamePlaceSkipReports>
        <testSamePlaceSkipReports> [0..1]
            <mode>{Enum}</mode>
            <radius>{Integer}</radius>
            <cyclesBeforeSkipping>{Integer}</cyclesBeforeSkipping>
            <cyclesToSkip>{Integer}</cyclesToSkip>
        </testSamePlaceSkipReports>
        <emergencySamePlaceSkipReports> [0..1]
            <mode>{Enum}</mode>
            <radius>{Integer}</radius>
            <cyclesBeforeSkipping>{Integer}</cyclesBeforeSkipping>
            <cyclesToSkip>{Integer}</cyclesToSkip>
        </emergencySamePlaceSkipReports>
        <deliveryShortCode>{Integer}</deliveryShortCode> [0..1]
        <outputPinsSetup> [0..1]
            <signalingPins>{Integer}</signalingPins>
            <ignoreTestAndEmergencyPins>{Boolean}</ignoreTestAndEmergencyPins>
        </outputPinsSetup>
        <signalingRepetitions>{Integer}</signalingRepetitions> [0..1]
        <blockInvalidGpsReports>{Enum}</blockInvalidGpsReports> [0..1]
        <emergencyReportFlood>{Integer}</emergencyReportFlood> [0..1]
        <outputPinStates>{Integer}</outputPinStates> [0..1]
    </request>
</updateResponse2>
```

ELEMENT(S)	DESCRIPTION
status	Values: Success Invalid Password Old Date Time Stamp
normalCallable testCallable emergencyCallable	Values: Off On On While in Motion
blockInvalidGpsReports	Values: No Blocking All Modes Normal and Test Modes Normal and Emergency Modes Only Normal Mode
normalSamePlaceSkipReports/mode testSamePlaceSkipReports/mode emergencySamePlaceSkipReports/mode	Values: No Skipping Skip Specified Cycles Skip Until Motion
outputPinStates	Each bit represents the state of an output pin.
inputPinStates	Each bit represents the state of an input pin.
signalingPins	Each bit corresponds to an output pin.

Update Response 3

```
<updateResponse3>
    <validation>{Enum}</validation>
    <results>
        <addCallOutSchedule>{Enum}</addCallOutSchedule> [0..*]
        <addGeofence>{Enum}</addGeofence> [0..*]
        <awakeTimeBetweenReports>{Enum}</awakeTimeBetweenReports> [0..*]
        <awakeTimeToKeepTrying>{Enum}</awakeTimeToKeepTrying> [0..*]
        <blockInvalidGpsReports>{Enum}</blockInvalidGpsReports> [0..*]
        <blockInvalidGpsReportsPerMode>{Enum}</blockInvalidGpsReportsPerMode> [0..*]
        <bufferAndSendMissedReports>{Enum}</bufferAndSendMissedReports> [0..*]
        <callable>{Enum}</callable> [0..*]
        <clearLastDateTime>{Enum}</clearLastDateTime> [0..*]
        <dataCallDestination>{Enum}</dataCallDestination> [0..*]
        <dataLoggingEnabled>{Enum}</dataLoggingEnabled> [0..*]
        <emergencyAcknowledgement>{Enum}</emergencyAcknowledgement> [0..*]
        <emergencyAwakeTimeBetweenReports>{Enum}</emergencyAwakeTimeBetweenReports> [0..*]
        <emergencyAwakeTimeToKeepTrying>{Enum}</emergencyAwakeTimeToKeepTrying> [0..*]
        <emergencyCallable>{Enum}</emergencyCallable> [0..*]
        <emergencyEnabled>{Enum}</emergencyEnabled> [0..*]
        <emergencyMailboxChecksBetweenReportCycles>{Enum}</emergencyMailboxChecksBetweenReportCycles> [0..*]
        <emergencyMotionSensorWait>{Enum}</emergencyMotionSensorWait> [0..*]
        <emergencyReportFlood>{Enum}</emergencyReportFlood> [0..*]
        <emergencySamePlaceSkipReports>{Enum}</emergencySamePlaceSkipReports> [0..*]
        <emergencyTimeBetweenReports>{Enum}</emergencyTimeBetweenReports> [0..*]
        <emergencyTimeToKeepTrying>{Enum}</emergencyTimeToKeepTrying> [0..*]
        <generalProfileHeader>{Enum}</generalProfileHeader> [0..*]
        <geofenceCheckFrequency>{Enum}</geofenceCheckFrequency> [0..*]
        <gsmSendMethod>{Enum}</gsmSendMethod> [0..*]
        <gsmTcpConnectionType>{Enum}</gsmTcpConnectionType> [0..*]
        <gsmTcpDestination>{Enum}</gsmTcpDestination> [0..*]
        <gsmTcpTimeOuts>{Enum}</gsmTcpTimeOuts> [0..*]
        <ignoreTestAndEmergencySwitches>{Enum}</ignoreTestAndEmergencySwitches> [0..*]
        <includeGpsInMessages>{Enum}</includeGpsInMessages> [0..*]
        <inputPin>{Enum}</inputPin> [0..*]
        <links>{Enum}</links> [0..*]
        <linkSwitchTime>{Enum}</linkSwitchTime> [0..*]
        <mailboxCheckRate>{Enum}</mailboxCheckRate> [0..*]
        <mailboxChecksBetweenReportCycles>{Enum}</mailboxChecksBetweenReportCycles> [0..*]
        <modifyGeofence>{Enum}</modifyGeofence> [0..*]
        <motionSensorAwakes>{Enum}</motionSensorAwakes> [0..*]
        <motionSensorAwakesPerMode>{Enum}</motionSensorAwakesPerMode> [0..*]
        <motionSensorBegin>{Enum}</motionSensorBegin> [0..*]
        <motionSensorBeginWithWindowSize>{Enum}</motionSensorBeginWithWindowSize> [0..*]
        <motionSensorEnd>{Enum}</motionSensorEnd> [0..*]
        <motionSensorWait>{Enum}</motionSensorWait> [0..*]
        <outputPinStates>{Enum}</outputPinStates> [0..*]
        <pollAntennaStatus>{Enum}</pollAntennaStatus> [0..*]
        <pollEmergencyActivationSource>{Enum}</pollEmergencyActivationSource> [0..*]
        <pollGsmLastReportTimes>{Enum}</pollGsmLastReportTimes> [0..*]
```

```
        <pollIridiumLastReportTimes>{Enum}</pollIridiumLastReportTimes> [0..*]
        <pollReport>{Enum}</pollReport> [0..*]
        <pollStatistics>{Enum}</pollStatistics> [0..*]
        <powerUpDelay>{Enum}</powerUpDelay> [0..*]
        <powerUpTimeout>{Enum}</powerUpTimeout> [0..*]
        <queuedReports>{Enum}</queuedReports> [0..*]
        <remoteUpdateTimeCheckEnabled>{Enum}</remoteUpdateTimeCheckEnabled> [0..*]
        <removeCallOutSchedule>{Enum}</removeCallOutSchedule> [0..*]
        <removeGeofence>{Enum}</removeGeofence> [0..*]
        <reportFlood>{Enum}</reportFlood> [0..*]
        <reportFormat>{Enum}</reportFormat> [0..*]
        <responseMethod>{Enum}</responseMethod> [0..*]
        <samePlaceSkipReports>{Enum}</samePlaceSkipReports> [0..*]
        <signalingPins>{Enum}</signalingPins> [0..*]
        <signalingRepetitions>{Enum}</signalingRepetitions> [0..*]
        <smsDestination>{Enum}</smsDestination> [0..*]
        <smsEmailDestination>{Enum}</smsEmailDestination> [0..*]
        <smsPhoneNumberDestination>{Enum}</smsPhoneNumberDestination> [0..*]
        <smsValidityPeriod>{Enum}</smsValidityPeriod> [0..*]
        <startupProfile>{Enum}</startupProfile> [0..*]
        <successfulSendRequired>{Enum}</successfulSendRequired> [0..*]
        <testAwakeTimeBetweenReports>{Enum}</testAwakeTimeBetweenReports> [0..*]
        <testAwakeTimeToKeepTrying>{Enum}</testAwakeTimeToKeepTrying> [0..*]
        <testCallable>{Enum}</testCallable> [0..*]
        <testMotionSensorWait>{Enum}</testMotionSensorWait> [0..*]
        <testReportFlood>{Enum}</testReportFlood> [0..*]
        <testSamePlaceSkipReports>{Enum}</testSamePlaceSkipReports> [0..*]
        <testTimeBetweenReports>{Enum}</testTimeBetweenReports> [0..*]
        <testTimeToKeepTrying>{Enum}</testTimeToKeepTrying> [0..*]
        <time>{Enum}</time> [0..*]
        <timeBetweenReports>{Enum}</timeBetweenReports> [0..*]
        <timeToKeepTrying>{Enum}</timeToKeepTrying> [0..*]
        <trackingEnabled>{Enum}</trackingEnabled> [0..*]
        <trackingMethod>{Enum}</trackingMethod> [0..*]
        <trackingProfileHeader>{Enum}</trackingProfileHeader> [0..*]
        <unknown tagTable="" tagValue="">{Enum}</unknown> [0..*]
    </results>
</updateResponse3>
```

ELEMENT	DESCRIPTION
validation	Values: Valid Invalid Password Invalid Checksum Length Error
results/*	Values: Success Out of Range Length Error Invalid Profile Header Not Supported No Change Failed

Status Report 0

Status Report 0

```
<statusReport0>
    <antennaStatus>{Integer}</antennaStatus> [0..1]
    <emergencyActivationSource>{Enum}</emergencyActivationSource> [0..1]
    <gsmLastReportTimes> [0..1]
        <attempted>{DateTime}</attempted>
        <successful>{DateTime}</successful>
    </gsmLastReportTimes>
    <iridiumLastReportTimes> [0..1]
        <attempted>{DateTime}</attempted>
        <successful>{DateTime}</successful>
    </iridiumLastReportTimes>
    <overallGsmGpsStatistics> [0..1]
        <attempts>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </attempts>
        <successes>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </successes>
    </overallGsmGpsStatistics>
    <overallGsmSignalStatistics> [0..1]
        <attempts>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </attempts>
        <successes>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </successes>
    </overallGsmSignalStatistics>
    <overallIridiumGpsStatistics> [0..1]
        <attempts>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
```

```
            <threeD>{Integer}</threeD>
        </attempts>
        <successes>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </successes>
    </overallIridiumGpsStatistics>
    <overallIridiumSignalStatistics> [0..1]
        <attempts>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </attempts>
        <successes>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </successes>
    </overallIridiumSignalStatistics>
    <overallStatisticsTimestamps> [0..1]
        <firstValid>{DateTime}</firstValid>
        <lastValid>{DateTime}</lastValid>
    </overallStatisticsTimestamps>
    <trackingGsmGpsStatistics> [0..1]
        <attempts>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </attempts>
        <successes>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </successes>
    </trackingGsmGpsStatistics>
    <trackingGsmSignalStatistics> [0..1]
        <attempts>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </attempts>
```

```
        <successes>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </successes>
    </trackingGsmSignalStatistics>
    <trackingIridiumGpsStatistics> [0..1]
        <attempts>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </attempts>
        <successes>
            <noOrDROrTimeOnly>{Integer}</noOrDROrTimeOnly>
            <twoD>{Integer}</twoD>
            <threeD>{Integer}</threeD>
        </successes>
    </trackingIridiumGpsStatistics>
    <trackingIridiumSignalStatistics> [0..1]
        <attempts>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </attempts>
        <successes>
            <zeroOrOne>{Integer}</zeroOrOne>
            <twoOrThree>{Integer}</twoOrThree>
            <fourOrFive>{Integer}</fourOrFive>
        </successes>
    </trackingIridiumSignalStatistics>
    <trackingStatisticsTimestamps> [0..1]
        <firstValid>{DateTime}</firstValid>
        <lastValid>{DateTime}</lastValid>
    </trackingStatisticsTimestamps>
    <unknown tagTable="{Integer}" tagValue="{Integer}">{HexString}</unknown> [0..1]
</statusReport0>
```

ELEMENT	DESCRIPTION
emergencyActivationSource	Values: Latching Pin Momentary Pin Remote Update Base Station None